#12
LDJ
1-14-02

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of

William J. BAER et al.

Application No.: 09/219,934

Confirmation No.: 4144

Filed: December 23, 1998

Express Mail No. EF279045854US

Group Art Unit: 2155

Examiner: O. Duong

FOR: METHOD AND APPARATUS FOR USING CLASSES, ENCAPSULATING DATA WITH ITS BEHAVIORS, FOR TRANSFERRING BETWEEN DATABASES AND CLIENT APPLICATIONS AND FOR ENABLING APPLICATIONS TO ADAPT TO SPECIFIC CONSTRAINTS OF THE DATA

APPELLANTS' BRIEF ON APPEAL UNDER 37 C.F.R. §1.192

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Appellants, within a two (2) month period from the November 2, 2001, filing date of the Notice of Appeal, herein file an Appeal Brief drafted in accordance with the provisions of 37 C.F.R. §1.192, as follows:

I. REAL PARTY IN INTEREST

The real party in interest here is the owner of the application, IBM Corporation.

II. RELATED APPEALS AND INTERFERENCES

Appellants appealed on November 21, 2001 from the final Office Action dated August 22, 2001 in application No. 09/220,293 (hereinafter "the '293 application"). Appellants will also appeal from the final Office Action dated December 12, 2001 in application No. 09/220,291 (hereinafter "the '291 application"). Both the '293 application and the '291 application have the same specification as the present application, and were filed on the same day as the present

application. The present application was rejected by the Examiner, who construes "transferring data to a data store" in both Mullins and the present application as "querying." However, the '291 application and the '293 application were rejected by another Examiner, who construes "transferring data to a data store" in Mullins and the '293 application, but not the '291 application, as "writing data to a data store". To the best of their knowledge, Appellants are not aware of any interferences involving the present application.

III. STATUS OF CLAIMS

Claims 1-23 are pending in the present application. Claims 1-7, 9-13, 15-19, and 21-23 stand rejected under 35 U.S.C. §102(e) as being anticipated by USP 5,857,197 (Mullins), and claims 8, 14 and 20 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Mullins in view of USP 6,006,230 (Ludwig et al.)

IV. STATUS OF AMEDMENTS

The claims have not been amended pursuant to final rejection.

In preparing this Brief on Appeal, Appellants noticed that claim 3, rather than claim 4, was mistakenly typed as the dependency of claim 5. Accordingly, claim 5 in the Appendix of this Brief on Appeal has been amended to correct its dependency. Appellants also are filing this Amendment separately. This amendment does not raise any new issues. Appellants respectfully request entry and consideration of this amendment.

V. SUMMARY OF THE INVENTION

The present application provides a flexibly adaptable asset management system with read-write capability features for processing and manipulating assets. An asset of the present

application is defined to be a set of related data, or meta data, for a document or an object and/or the actual data itself (page 5, lines 6-7). The set of related data may consist of relational data, files, references to files from indexing engines, or any other combination of data types (page 5, lines 9-10). The assets may represent, for example, data in the form of text, full-motion video, audio, graphics, or images (page 2, lines 14-15).

The system comprises three layers: a client application layer, for manipulating and browsing assets; an asset manager server layer; and a data store representing the third layer of the system. The asset manager server layer comprises several configurable modules, including a client adapter module, a schema adapter module, and a resources module. The asset manager server provides, for specified asset types, programming interface services, such as storing, querying, and retrieving assets representing data to and from the data store. Each schema adapter maps the data store schema into assets of a particular asset type, and new schema adapters are constructed to support database operations for new types of asset (page 13, lines 13-16, and page 14, lines 6-7).

The schema adapter (used interchangeably with "schema controller" in the specification) defines the communication between the asset manager server and the data store, and is also responsible for implementing the interfaces defined by the client adapter, e.g., query, result, edit and system administrator (page 14, lines 4-6). Information shared between the client application and schema adapter can be either returning data to a client application or to update or add data to the data store (page 19, lines 29-31). In this way, the schema adapter of the present application transfers data to and from the data store in response to methods invoked in the client adapter by

the client application, as recited in claims 1-23. Users can add data to the data store, and update, delete, retrieve and query data in the data store. The system of the present application can also be used to transfer information between data stores. Therefore, "transferring data to a data store" in the present application means "writing."

The prior art provides a system for querying databases located on multiple platforms, processing data obtained, and presenting processed data to a user. The prior art, however, fails to disclose a flexibly adaptable asset management system with read-write capability features of the present application.

VI. ISSUES

1. Does the prior art teach or reasonably suggest the flexibly adaptable asset management system with read-write capability features, as recited in claims 1, 6, 12 and 18?

2. Does the prior art teach or reasonably suggest the provision of a schema adapter that is specific to a particular one of the assets, as recited in claims 2, 7, 13 and 19?

3. Does the prior art teach or reasonably suggest external services for providing a link between the at least one schema adapter and the data store for transferring data to and from the data store, as recited in claims 3, 9, 15 and 21?

4. Does the prior art teach or reasonably suggest the calling of a specific template for flexibly adaptable object oriented class(s), as recited in claims 4, 10, 16 and 22?

5. Does the prior art teach or reasonably suggest the calling of a specific template for the object oriented class(s) using an data type and an action path, as recited in claims 5, 11, 17 and 23?

6. Does the prior art teach or reasonably suggest the creating a local copy of the instance of the new object oriented class in the client application in a method of flexibly adapting asset management system, as recited in claims 8, 14 and 20?

VII. GROUPING OF CLAIMS

Claims 1, 6, 12 and 18 stand and fall together. Claims 2, 7, 13 and 19 stand and fall together. Claims 3, 9, 15 and 21 stand and fall together. Claims 4, 10, 16 and 22 stand and fall together. Claims 5, 11, 17 and 23 stand and fall together. Claims 8, 14 and 20 stand and fall together.

VIII. ARGUMENT

Issue 1

Introduction

The present invention significantly differs from Mullins. The Mullins system and method cannot be used for the claimed purpose of the present application, because Mullins only teaches reading data from a data store, but and not writing data to the data store.

After a telephone interview with Appellants on September 17, 2001, the Examiner agreed that Mullins discloses a read-only system, distinguished from the asset management system with read-write capability features disclosed in the present application, and that "transferring data to a data store" in Mullins means "querying".

However, the Examiner continues to maintain that "transferring data to a data store" as recited in the claims of the present application is sufficiently broad to encompass "querying" and that the read-write capability features therefore are not recited sufficiently in the claims. The

Examiner's interpretation of "transferring data to a data store" in the present application is different from that of the Examiner in the '293 application and the '291 application, who construes the term in the '293 applications, having the same specifications as the present application, as meaning "writing data to a data store".

Mullins

Mullins discloses a method and system for querying data stored in data stores, which could be either an object or a non-object data store, as objects from an object application. Specifically, Mullins seeks to solve a data store access problem which arises when two or more data stores, which have different underlying structures, are accessed for stored data. For example, in the system shown in Fig. 1 of Mullins, an object application 101 accesses both object data store 312 and non-object data store 302 for stored data.

The Mullins system comprises an adapter abstraction layer 600 which further comprises a first adapter 400 and a second adapter 500. The adapter abstraction layer 600 performs the necessary conversions between object and non-object views, and consequently allows both object data store and non-object data store to be accessed identically for stored data from at least one object application.

In describing this system, Mullins states that a request 100 and an accompanying object 102 are passed from an application program to the first adapter 400. The first adapter 400 then extracts object attributes 103 and the object name 104 from the object 102, packs object attributes 103 and the object name 104 as "data" 105, and communicates the "data" 105 and the request 100 to the second adapter 500 (Mullins at col. 7, lines 39-54). The second adapter 500

searches a meta data map using the object name 104, and generates a command 303, using the object attributes 103, for accessing the data store 302 according to the request 100 (Mullins at col. 7, lines 55-67). The second adapter 500 then executes command 303, obtains and processes the data store content 304 from data store 302, packs the obtained data store content 304 and the execution status as data 115, and communicates the data 115 to the first adapter 400 (Mullins at col. 8, lines 1-26).

Mullins names what is communicated from the first adapter to the second adapter as "data", but looking at this so-called "data" more closely, one can see that this "data is basically a command or a query, dramatically distinguished from "data" as described in the present application, which consists relational data, files, references to files from indexing engines, or any other combination of data types (page 5, lines 9-10).

It is clear that nothing is written to Mullins' data stores through the communication from the first adapter to the second adapter, and then to the data stores. In addition, Mullins specifically states that the use of its technology provides "read only" data stores over the Internet (Mullins at col. 7, lines 64-66).

Although "transferring the requested data store content from the first adapter" is mentioned in Mullins (Mullins at col. 4, lines 49-65), this transfer is not described in any detail in Mullins' specification, and moreover is contradicted by other discussion in Mullins, for example, at the bottom of column 7 of Mullins, which talks about a "read-only" data store. Thus, Appellants submit that the Mullins description on which the Examiner relies is not enabling.

Therefore, Mullins teaches a read-only system for reading data stored in data stores which have different underlying structures, but lacks disclosure of read-write capability features of the present application. Communication of data from the first adapter to the second adapter is not a transfer, as disclosed and claimed in the present application. "Transferring data to a data store" in the present application means "writing data to a data store". The ordinarily skilled artisan would interpret "transferring" data "to" a place, such as a data store, as writing data to the data store. The ordinarily skilled artisan would not interpret the transfer of data to a data store as broadly as to encompass querying. The Mullins system and method thus are no more than what Appellants acknowledge as prior art.

The Inherent Read-Write Capability Features

The read-write capability features are inherent in claims of the present application. Each of independent claims 1, 6, 12 and 18 clearly and specifically recites a schema adapter for mapping the assets to the data stored in the data store and for transferring the data to and from the data store in response to methods invoked by the client application. The set of related data may consist of relational data, files, references to files from indexing engines, or any other combination of data types (page 5, lines 9-10). It is clear that transferring such data to some location (as opposed to some command or query, which is what Mullins talks about) means writing that data to that place and that transferring such data from some location means reading that data from that place. Read in light of the specification, "transferring data to and from the data store" recited in the claims of the present application means "writing data to a data store and

reading data from a data store." Thus, the read-write capability features are intrinsically incorporated in the claims of the present application.

From the foregoing, it is clear that Mullins discloses at most "querying" by communicating a command. Moreover, Mullins discloses a read-only system. Pursuant to the foregoing discussion, Appellants submit that application claims 1, 6, 12 and 18 (and their dependencies) are patentable.

Issue 2

Appellants submit that, for all the reasons which have been discussed in detail above, claims 2, 7, 13 and 19 are patentable at least by virtue of their respective dependence on the patentable independent claims 1, 6, 12 and 18.

Furthermore, Mullins discloses a method and system for querying data stored in data stores having different underlying structures, e.g., object data store 312 and non-object data store 302. The adapter abstraction layer 600 performs the necessary conversions between object and non-object views, and therefore allows both object data store and non-object data store to be accessed identically from at least one object application (Mullins, col. 4, lines 9-14). The abstraction layer 600 is applicable to data stores having different data structures, and the adapters can be transparently interchanged by the object application (Mullins, col. 9, lines 34-38). Mullins does not teach a schema adapter that is specific to a particular one of the assets, as claimed in the present application. Accordingly, Appellants submit that claims 2, 7, 13 and 19 are patentable for this additional reason as well.

Issue 3

Appellants submit that, for all the reasons which have been discussed in detail above, claims 3, 9, 15 and 21 are patentable at least by virtue of their respective dependence on the patentable independent claims 1, 6, 12 and 18.

Furthermore, Mullins teaches a read-only system for querying data stored in data stores having different underlying structures. Nothing is written to Mullins' data stores through the communication from the first adapter to the second adapter, and then to the data stores. What is communicated along the link between the data stores and the second adapter 500 of the Mullins system includes commands from the second adapter 500 and stored data from the data stores, but does not include assets to be written to the data stores as claimed in the present application. Therefore, Appellants submit that claims 3, 9, 15 and 21 are patentable for this additional reason as well.

Issue 4

Appellants submit that, for all the reasons which have been discussed in detail above, claims 4, 10, 16 and 22 are patentable at least by virtue of their respective dependence on the patentable independent claims 1, 6, 12 and 18.

Furthermore, Mullins seeks to solve a data store access problem which arises when two or more data stores, which have different underlying structures, are accessed for stored data. The second adapter 500 of the Mullins system transfers the requested data store content to the first adapter 400. As each object data is being read from the data stores, an instance of the object's class type (e.g., a Java class) is instantiated and initialized to the row of attributes returned in the

result set (Mullins, at col. 8, lines 21-26). Each such java class must implement a standard interface, so that the system can call the methods necessary to initialize the data automatically to make new instances automatic and transparent to the client object application (Mullins, at col. 8, lines 30-36).

In contrast, each schema adapter of the present application maps the data store schema into assets of a particular asset type and a new schema adapter is constructed to support database operations for a new type of asset (page 13, lines 13-16, and page 14, lines 6-7). The plug-in templates are dynamically loaded, and the set of plug-in templates is dependent on which schema adapter are loaded and what plug-ins are needed by the loaded schema adapter (page 20, lines 5-7).

In short, each java class in the Mullins system implements a standard interface, while the system of the present application calls a specific template for the flexibly adaptable object oriented class(s), depending on plug-ins needed by the loaded schema adapter corresponding to a particular asset type. Mullins does not teach or even suggest the calling of the specific template for the flexibly adaptable object oriented class(s). Therefore, Appellants submit that claims 4, 10, 16 and 22 are patentable for this additional reason as well.

Issue 5

Appellants submit that, for all the reasons which have been discussed in detail above, claims 5, 11, 17 and 23 are patentable at least by virtue of their respective dependence from the patentable independent claims 1, 6, 12 and 18. Claims 5, 11, 17 and 23 are also patentable by virtue of their respective dependence from the patentable dependent claims 4, 10, 16, and 22.

Furthermore, the plug-in template in the present application is specified using a domain and an action path. The domain defines the data type the plug-in supports, and the action path defines the context of the plug-in (page 24, lines 4-6). However, each java class in the Mullins system must implement a standard interface. Mullins does not teach the calling of the specific template for the flexibly adaptable object oriented class(s). Even assuming *arguendo* that Mullins teaches the calling of the specific template for the flexibly adaptable object oriented class(s), it does not teach or suggest the use of data type and an action path for such calling. Therefore, Appellants submit that claims 5, 11, 17 and 23 are patentable for this additional reason as well.

Issue 6

As discussed above, Mullins does not teach a flexibly adaptable asset management system with read-write capability features disclosed and claimed in the present application. Accordingly, claims 8, 14 and 20 are patentable at least by virtue of their respective dependence on the patentable independent claims 6, 12 and 18.

The Examiner has asserted that claims 8, 14 and 20 are unpatentable over Mullins in view of Ludwig et al., which teaches a database client/server development system with the limitations of "creating a copy of the object instance at the particular client". Even assuming *arguendo* that Mullins could be construed as teaching the read-write capability features of the present application, Ludwig et al. does not teach "creating a local copy of the instance of the new object oriented class in the client application" recited in the present application.

Ludwig et al.

Ludwig et al. discloses a method for partitioning application objects among multiple computers in a distributed computing environment comprising one or more clients 210 connected to one or more servers 230. In response to a request to partition application objects, Ludwig et al. system creates an object instance of a particular object at a particular server and a corresponding proxy object for the particular object at a particular client. The proxy object defines properties for the object instance for the particular object and an interface for executing methods available to the object instance (Ludwig et al., col. 39, lines 11-18). When a *SetConnect* call 515 associates the proxy object with the server and consequently leads to the invocation of *CreateRemoteObject* handler 535, the system attempts to instantiate the corresponding object at the server side. When the real object is instantiated at the server, it is created with the same properties and methods for which corresponding stubs were created at the proxy (Ludwig et al., col. 10, lines 45-52) . In this way, a non-visual object is created at the client.

In contrast, the schema adapter 707 calls the *setPlug-inValue* and *setDomain* method on the remote plug-in container 709. The remote plug-in container 709 returns the remote plug-in container 709 to the client adapter 705. The client adapter 705 calls the *createLocalPI* method to construct the local plug-in on the client application (page 27, lines 8-12).

It is clear that, read in light of specification, the method for creating a copy of object instance at a particular client in Ludwig et al. and the method for creating a local copy of the instance of the new object oriented class in the present application significantly differ from each

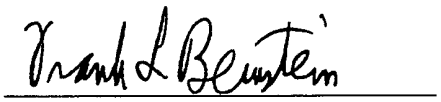
other: In Ludwig's method, a proxy object and a call associates the proxy object with the server, while in the present method an instance of the object oriented class is returned to the client application. Therefore, Ludwig et al. does not teach or suggest the method of creating a local copy of the instance of the new object oriented class in the present application. Appellants submit that claims 8, 14 and 20 in the subject application are patentable for this additional reason as well.

IX. CONCLUSION

Pursuant to the foregoing arguments, Appellants submit that claims 1-23 in the subject application are patentable.

The present Brief on Appeal is being filed in triplicate. Appellants hereby petition for any extension of time which may be required to maintain the pendency of this case, and any required fee for such extension is to be charged to Deposit Account No. 19-4880.

Respectfully submitted,



Frank L. Bernstein
Registration No. 31,484

SUGHRUE MION, PLLC
1010 El Camino Real, Suite 360
Menlo Park, CA 94025

Tel: 650-325-5800
Fax: 650-325-6606

Date: December 31, 2001

APPENDIX

CLAIMS 1-23 ON APPEAL:

1. A flexibly adaptable asset management system for deploying asset management functions to a client application for manipulating assets, representing data, in a data store, using classes for transfers between the data store and the client application, the system comprising:

an asset manager server disposed between the client application and the data store, the asset manager server including:

at least one client adapter for providing interface functions between the client application and the asset manager server;

at least one schema adapter for mapping the assets to the data stored in the data store and for transferring the data to and from the data store in response to methods invoked in the at least one client adapter by the client application; and

at least one object oriented class, being one of the classes, wherein an instance of the at least one object oriented class encapsulates the data and associated behaviors for transferring between the at least one schema adapter and the client application through the at least one client adapter,

wherein, the at least one object oriented class is flexibly adaptable, thereby allowing the system to do one or more of handle different data types and associated behaviors and handle additional client applications.

2. The system according to claim 1, wherein the at least one schema adapter is specific to a particular one of the assets, an asset being meta data for a particular data type.

3. The system according to claim 1, wherein the asset manager server further comprises external services for providing a link between the at least one schema adapter and the data store.

4. The system according to claim 1, wherein the at least one schema adapter:

calls a specific template for the at least one object oriented class;

produces the instance of the at least one object oriented class from the template; and

initializes the instance of the object oriented class prior to the transferring between the at least one schema adapter and the client application through the at least one client adapter.

5. The system according to claim 4, wherein the at least one schema adapter calls the specific template for the at least one object oriented class using the data type and an action path provided to the at least one schema adapter from the client application through the at least one client adapter.

6. A method of flexibly adapting asset management system for deploying asset management functions to a client application for manipulating assets, representing data, in a data store, using classes for transfers between the data store and the client application, the system comprising:

an asset manager server disposed between the client application and the data store, the asset manager server including:

at least one client adapter for providing interface functions between the client application and the asset manager server;

at least one schema adapter for mapping the assets to the data stored in the data store and for transferring the data to and from the data store in response to methods invoked in the at least one client adapter by the client application; and

at least one object oriented class, being one of the classes, wherein an instance of the at least one object oriented class encapsulates the data and associated behaviors for transferring between the at least one schema adapter and the client application through the at least one client adapter,

wherein, the at least one object oriented class is flexibly adaptable, thereby allowing the system to do one or more of handle different data types and associated behaviors and handle additional client applications,

the method comprising creating a new object oriented class by:

choosing a template for the new object oriented class;

choosing a domain for an instance of the new object oriented class; and

implementing methods for retrieving and setting values for the instance of the new object oriented class.

7. The method according to claim 6, where the at least one schema adapter is specific to a particular one of the assets, an asset being meta data for a particular data type.

8. The method according to claim 6, further comprising the steps of:

creating a local copy of the instance of the new object oriented class in the client application; and implementing remote and local methods and interfaces to support the instance and the local copy of the instance respectively.

9. The method according to claim 6, wherein the asset manager server further comprises external services for providing a link between the at least one schema adapter and the data store.

10. The method according to claim 6, wherein the at least one schema adapter:

calls a specific template for the at least one object oriented class;

produces the instance of the at least one object oriented class from the specific template; and

initializes the instance of the object oriented class prior to the transferring between the at least one schema adapter and the client application through the at least one client adapter.

11. The method according to claim 10, wherein the at least one schema adapter calls the specific template for the at least one object oriented class using the data type and an action path provided to the at least one schema adapter from the client application through the at least one client adapter.

12. A program storage device readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps flexibly adapting an asset management system for deploying asset management functions to a client application for manipulating assets, representing data, in a data store, using classes for transfers between the data store and the client application, the system comprising:

an asset manager server disposed between the client application and the data store, the asset manager server including:

at least one client adapter for providing interface functions between the client application and the asset manager server;

at least one schema adapter for mapping the assets to the data stored in the data store and for transferring the data to and from the data store in response to methods invoked in the at least one client adapter by the client application; and

at least one object oriented class, being one of the classes, wherein an instance of the at least one object oriented class encapsulates the data and associated behaviors for transferring between the at least one schema adapter and the client application through the at least one client adapter,

wherein, the at least one object oriented class is flexibly adaptable, thereby allowing the system to do one or more of handle different data types and associated behaviors and handle additional client applications,

the method comprising creating new ones of the at least one object oriented class by:

choosing a template for the new object oriented class;

choosing a domain for an instance of the new object oriented class; and

implementing methods for retrieving and setting values for the instance of the new object oriented class.

13. The program storage device according to claim 12, wherein the at least one schema adapter is specific to a particular one of the assets, an asset being meta data for a particular data type.

14. The program storage device according to claim 12, further comprising the steps of:

creating a local copy of the instance of the new object oriented class in the client application; and

implementing remote and local methods and interfaces to support the instance and the local copy of the instance respectively.

15. The program storage device according to claim 12, wherein the asset manager server further comprises external services for providing a link between the at least one schema adapter and the data store.

16. The program storage device according to claim 12, wherein the at least one schema adapter:

calls a specific template for the at least one object oriented class;

produces the instance of the at least one object oriented class from the specific template; and

initializes the instance of the object oriented class prior to the transferring between the at least one schema adapter and the client application through the at least one client adapter.

17. The program storage device according to claim 16, wherein the at least one schema adapter calls the specific template for the at least one object oriented class using the data type and an action path provided to the at least one schema adapter from the client application through the at least one client adapter.

18. A system for flexibly adapting an asset manager for deploying asset management functions to a client application for manipulating assets, representing data, in a data store, using classes for transfers between the data store and the client application, the system comprising:

an asset manager server disposed between the client application and the data store, the asset manager server including:

at least one client adapter for providing interface functions between the client application and the asset manager server;

at least one schema adapter for mapping the assets to the data stored in the data store and for transferring the data to and from the data store in response to methods invoked in the at least one client adapter by the client application; and

at least one object oriented class, being one of the classes, wherein an instance of the at least one object oriented class encapsulates the data and associated behaviors for transferring between the at least one schema adapter and the client application through the at least one client adapter,

wherein, the at least one object oriented class is flexibly adaptable, thereby allowing the system to do one or more of handle different data types and associated behaviors and handle additional client applications, and

further wherein, a new object oriented class is created by:

choosing a template for the new object oriented class;

choosing a domain for an instance of the new object oriented class; and

implementing methods for retrieving and setting values for the instance of the new object oriented class.

19. The system according to claim 18, wherein the at least one schema adapter is specific to a particular one of the assets, an asset being meta data for a particular data type.

20. The system according to claim 18, wherein the new object oriented class is created by further:

creating a local copy of the instance of the new object oriented class in the client application; and

implementing remote and local methods and interfaces to support the instance and the local copy of the instance respectively.

21. The system according to claim 18, wherein the asset manager server further comprises external services for providing a link between the at least one schema adapter and the data store.

22. The system according to claim 18, wherein the at least one schema adapter:

calls a specific template for the at least one object oriented class;

produces the instance of the at least one object oriented class from the specific template; and

initializes the instance of the object oriented class prior to the transferring between the at least one schema adapter and the client application through the at least one client adapter.

23. The system according to claim 22, wherein the at least one schema adapter calls the specific template for the at least one object oriented class using the data type and an action

path provided to the at least one schema adapter from the client application through the at least one client adapter.